Probes

At times you need to gather information from a client machine before you can generate its configuration. For example, if some of your machines have both a local scratch disk and a system disk while others only have the system disk, you would want to know this information to correctly generate an `/etc/auto.master` autofs config file for each type. Here we will look at how to do this.

First you will need to set up the TCheetah plugin, as described on the TCheetahPlugin page.

Next, we need to create a `Probes` directory in our toplevel repository location:

```
mkdir /var/lib/bcfg2/Probes
```

This directory will hold any small scripts we want to use to grab information from client machines. These scripts can be in any scripting language; the shebang line (the `#!/usr/bin/env some_interpreter_binary` line at the very top of the script) is used to determine the script's interpreter.

Now we need to figure out what exactly we want to do. In this case, we want to hand out an `/etc/auto.master` file that looks like:

```
/software  /etc/auto.software --timeout 3600
/home      /etc/auto.home --timeout 3600
/hometest  /etc/auto.hometest --timeout 3600
/nfs       /etc/auto.nfs --timeout 3600
/scratch   /etc/auto.scratch --timeout 3600
```

for machines that have a scratch disk. For machines without an extra disk, we want to get rid of that last line:

```
/software  /etc/auto.software --timeout 3600
/home      /etc/auto.home --timeout 3600
/hometest  /etc/auto.hometest --timeout 3600
/nfs       /etc/auto.nfs --timeout 3600
```

So, from the Probes standpoint we want to create a script that counts the number of SCSI disks in a client machine. To do this, we create a very simple `Probes/scratchlocal` script:

```
cat /proc/scsi/scsi | grep Vendor | wc -l
```

Running this on a node with `n` disks will return the number `n+1`, as it also counts the controller as a device. To differentiate between the two classes of machines we care about, we just need to check the output of this script for numbers greater than 2. We do this in the template.

The `TCheetah/` directory is laid out much like the `Cfg/` directory. For this example we will want to create a `TCheetah/etc/auto.master` directory to hold the template of the file in question. Inside of this template we will need to check the result of the Probe script that got run and act accordingly. The `TCheetah/etc/auto.master/template` file looks like:

```
/software  /etc/auto.software --timeout 3600
/home      /etc/auto.home --timeout 3600
/hometest  /etc/auto.hometest --timeout 3600
/nfs       /etc/auto.nfs --timeout 3600
#if int($self.metadata.probes["scratchlocal"]) > 2
```

```
/scratch   /etc/auto.scratch --timeout 3600
#end if
```

Any Probe script you run will store its output in `$self.metadata.probes["scriptname"]`, so we get to our `scratchlocal` script's output as seen above. Note that we had to wrap the output in an `int()` call; the script output is treated as a string, so it needs to be converted before it can be tested numerically.

With all of these pieces in place, the following series of events will happen when the client is run:

1. Client runs
2. Server hands down our `scratchlocal` probe script
3. Client runs the `scratchlocal` probe script and hands its output back up to the server
4. Server generates `/etc/auto.master` from its template, performing any templating substitutions/actions needed in the process.
5. Server hands `/etc/auto.master` down to the client
6. Client puts file contents in place.

Now we have a nicely dynamic `/etc/auto.master` that can gracefully handle machines with different numbers of disks. All that's left to do is to add the `/etc/auto.master` to a Bundle:

```
<ConfigFile name='/etc/auto.master'/>
```